INTRODUCTION TO NODE.JS

NODE JS:

- Popular open source ,cross platform ,server side runtime environment that allows developers to build scalable , high performance applications.
- One of the key benefits of it is that it allows developers to use JS on Both front-end and back-end of an application
- Uses asynchronous programming -
    - **ASYNCHRONOUS PROGRAMMING:** An *asynchronous* model allows multiple things to happen at the same time. When you start an action, your program continues to run.
- Wide range of applications; large and active community of developers.

NPM - Node Package Manager:

- It is a command line tool that is used to install manage and share packages (i.e., libraries, modules, and other dependencies)

EXPRESS JS:

- Open -source framework that provides tools and features that simplify the development of web Applications and API's.
- Provides a light weight set of features that can be extended using middlewares.
- **ROUTING:** To handle GET , POST , PUT , DELETE requests
- **MIDDLE WARES:** Allows developers to use middleware functions to modify incoming requests and outgoing responses .
  Middleware functions can be used to handle authentication , logging , error handling etc
- **TEMPLATE ENGINES :** Used to generate dynamic HTML pages (eg. ,EJS)
- Provides built in error-handling mechanism.
- Works on top of Node.js webserver functionality to simplify its API.

BODY PARSER:

- Body Parser is a middleware of Node JS used to handle HTTP POST request. Body Parser can parse string based client request body into JavaScript Object which we can use in our application.

  Installation Command: npm install body-parser

NODEMON:

- nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

  Installation Command: npm install -g nodemon

LINE BY LINE CODE EXPLANATION

/*

To use any package that we have downloaded, we need to import them into our logic file (i.e., index.js file in this case) and needs to be assigned to a variable to access the packages functions and utilities

So basically require ('[package name]') imports the package and assign it to a variable

const [variable_name] = require ('[package name]')

***/**

```
const express = require("express")

const bodyParser = require("body-parser")
```

/*

The below line creates an instance of the express application, which implies that using the 'app' variable we can tap into the functionalities offered by express package such as get (), post (), listen () etc.

***/**

```
const app = express()
```

/*

The below line tells the express application to use the 'body-parser' middleware package to parse incoming requests with URL-encoded payloads, i.e., it allows us to access the data submitted by the user in the frontend.

When extended is set to true, the URL-encoded data can be a JSON-like object or an array. If it's set to false, the URL-encoded data can only be a string or an array.

Syntax : **app.use([variable_name].type_of_accessing())**

In this code we have assigned the body-parser package to ‘bodyParser’ variable, hence we used ‘bodyParser’ or for example, if we have assigned the body-parser as follows:

const parser = require(‘body-parser’)

then =>

app.use(parser.urlencoded({ extended:true }))

Other accessing methods include json() , text() etc

***/**

app.use(bodyParser.urlencoded({ extended: true }))

/*

This line sets up a route for handling GET requests to the root URL ('/'). When a GET request is made to the root URL, the server responds by sending the home.html file to the client and logging a message to the console.

An application can have multiple routes and we need to define each and every route in out ‘index.js’ file.

The get function takes two parameters: ‘**the route name**’ and ‘**a call-back function**’

The route name is the route that the client wants to target.

A call-back function is passed as an argument to another function and is executed after the main function has completed its operation. The main function can then use the result of the call-back function for further processing. The call-back function can then perform some action based on the result or data passed to it by the main function.

‘__dirname’ is a special global variable that refers to the absolute path of the directory that contains the currently executing file. ‘__dirname’ is used to construct the absolute path of the home.html file. By using ‘__dirname’, we ensure that the file is always located relative to the current file's directory, regardless of where the application is deployed or executed.

*/

```
app.get('/', function (req, res) {
   res.sendFile(__dirname + '/home.html')
   console.log('Susccefully accesed page')
})
```

/*

This line sets up a route for handling POST requests to the /calculate URL. When a POST request is made to the /calculate URL, the server retrieves the values of number1 and number2 from the request body, adds them together, and sends the result back to the client.

The ‘res.send()’ method in Node.js is used to send a response to an HTTP request. It can send various types of data as the response. The following are some of the parameters that can be sent using ‘res.send()’: a string, a buffer, a JSON object, a HTML string, an array of JSON objects, a boolean value

It's important to note that res.send() can only be called once per request. If it's called multiple times, it will result in an error. Therefore, it's recommended to use res.send() to send the final response and not to call it multiple times within the same request handler.

*/

```
app.post('/calculate', function (req, res) {
   var num1 = Number(req.body.number1)
   var num2 = Number(req.body.number2)
   var result = num1 + num2
   res.send("The result is " + result)
})
```

/*

This line starts the express application and listens for incoming HTTP requests on port 3000. When the server starts, it logs a message to the console to indicate that it is running.

*/

```
app.listen(3000, function () {
   console.log('Server started at port 3000');
})
```